

AP
JFW

Attorney's Docket No.: 9209-10

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re: Tabares et al.

Serial No.: 09/992,155

Filed: November 5, 2001

For: METHODS, SYSTEMS AND COMPUTER PROGRAM PRODUCTS FOR
INSTANTIATING A DEVICE DRIVER FOR COMMUNICATION WITH A DEVICE
BY DYNAMICALLY ASSOCIATING THE DEVICE DRIVER AT RUN-TIME WITH
A DEVICE-SPECIFIC AND/OR SERVICE-SPECIFIC SOFTWARE COMPONENT

Examiner: Diem K. Cao

Group Art Unit: 2194


Confirmation No.: 5291

Date: February 27, 2006

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Certificate of Mailing

I hereby certify that this correspondence is being deposited
with the United States Postal Service as first class mail in an
envelope addressed to: Mail Stop Appeal Brief-Patents,
Commissioner for Patents, P.O. Box 1450, Alexandria, VA
22313-1450, on February 27, 2006


Traci A. Brown

**TRANSMITTAL OF APPEAL BRIEF
(PATENT APPLICATION--37 C.F.R. § 1.192)**

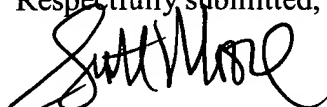
1. Transmitted herewith is the APPEAL BRIEF for the above-identified application,
pursuant to the Notice of Appeal received by the US Patent and Trademark Office on **December
27, 2005**.

2. Pursuant to 37 C.F.R. § 1.17(c), the fee for filing the Appeal Brief is:

<input type="checkbox"/>	small entity	\$250.00
<input checked="" type="checkbox"/>	other than small entity	\$500.00

3. ☒ Any additional fee or refund may be charged to Deposit Account
50-0220.

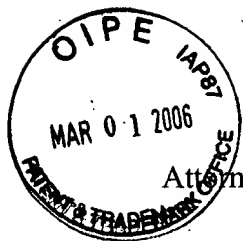
Respectfully submitted,



D. Scott Moore

Registration No. 42,011

Myers Bigel Sibley & Sajovec, P.A.
P. O. Box 37428
Raleigh, North Carolina 27627
Telephone: (919) 854-1400
Facsimile: (919) 854-1401
Customer No. 20792



Attorney's Docket No.: 9209-10

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re: Tabares et al.

Serial No.: 09/992,155

Filed: November 5, 2001

For: METHODS, SYSTEMS AND COMPUTER PROGRAM PRODUCTS FOR
INSTANTIATING A DEVICE DRIVER FOR COMMUNICATION WITH A DEVICE
BY DYNAMICALLY ASSOCIATING THE DEVICE DRIVER AT RUN-TIME WITH
A DEVICE-SPECIFIC AND/OR SERVICE-SPECIFIC SOFTWARE COMPONENT

Examiner: Diem K. Cao

Group Art Unit: 2194


Confirmation No.: 5291

Date: February 27, 2006

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Certificate of Mailing

I hereby certify that this correspondence is being deposited
with the United States Postal Service as first class mail in an
envelope addressed to: Mail Stop Appeal Brief-Patents,
Commissioner for Patents, P.O. Box 1450, Alexandria, VA
22313-1450, on February 27, 2006


Traci A. Brown

APPELLANTS' BRIEF ON APPEAL UNDER 37 C.F.R. §41.37

Sir:

This Appeal Brief is filed pursuant to the "Notice of Appeal to the Board of Patent Appeals and Interferences" mailed December 22, 2005 and received in the U. S. Patent and Trademark Office on December 27, 2005.

Real Party In Interest

The real party in interest is assignee Trendium, Inc., Fort Lauderdale, Florida.

Related Appeals and Interferences

Appellants are aware of no appeals or interferences that would be affected by the present appeal.

Status of Claims Status of Claims

Appellants appeal the final rejection of Claims 1 - 6, 9 - 15, 20 - 25, 28 - 34, 39 - 44, and 47 - 53, which as of the filing date of this Brief remain under consideration. The claims at issue as included in Appellants' response to the Office Action of April 20, 2005 are attached hereto as Appendix A.

Status of Amendments

Two responses have been filed in the present case: A "Request for Reconsideration" in which no claims were amended was filed November 2, 2004 in response to an Office Action mailed August 2, 2004 (hereinafter "First Action"). An "Amendment" was filed July 20, 2005 in response to an Office Action mailed April 20, 2005 (hereinafter "Second Action"). A final Office Action was mailed September 26, 2005. Appellants have canceled Claims 16 - 19, 35 - 38, and 54 - 57 in prosecuting the present application; therefore, Claims 1 - 15, 20 - 34, and 39 - 53 remain for consideration on the present appeal.

Summary of Claimed Subject Matter

Appellants appeal the final rejection of Independent Claims 1, 12, 20, 31, 39, and 50.

Independent Claim 1 is directed to a computer implemented method of instantiating a device driver in which a first software component is dynamically associated with the device driver at run-time. The first software component contains information that facilitates communication with devices of a specific device type. (Specification, page 10, lines 13 - 18; FIG. 6).

Independent Claim 12 is directed to a computer implemented method of collecting data from a device. A request to collect data is received from the device. (Specification, page 11, lines 15 - 16; FIG. 9, block 132). A software component is dynamically associated with a device driver at run-time. The software component contains information that facilitates communication with the device. (Specification, page 11, lines 21 - 24; FIG. 9, block 134). Data is retrieved from the device using the device driver. (Specification, page 11, lines 24 - 25; FIG. 9, block 136).

Independent Claim 20 is directed to a system for instantiating a device driver that includes means for dynamically associating a first software component with the device driver at run-time. The first software component contains information that facilitates communication with devices of a specific device type. (Specification, page 10, lines 13 - 18; FIG. 6). The device driver manager 86, processor 72, and memory 74 of FIG. 3 provide structure for the means for dynamically associating.

Independent Claim 31 is directed to a system for collecting data from a device that includes means for receiving a request to collect data from the device. (Specification, page 11, lines 15 - 16; FIG. 9, block 132). In addition, the system includes means for dynamically associating a software component with a device driver at run-time. The software component contains information that facilitates communication with the device. (Specification, page 11, lines 21 - 24; FIG. 9, block 134). The system further includes means for retrieving data from the device using the device driver. (Specification, page 11, lines 24 - 25; FIG. 9, block 136). The access device program 84, processor 72, and memory 74 of FIG. 3 provide structure for the means for receiving. The device driver manager 86, processor 72, and memory 74 of FIG. 3 provide structure for the means for dynamically associating. The service management system 24 of FIG. 1 provides structure for the means for retrieving.

Independent Claim 39 is directed to a computer program product for instantiating a device driver in which a computer readable storage medium has computer readable program code embodied therein. (Specification, page 4, line 14 - page 5, line 3, and page 9, line 17 - page 10, line 8). The computer readable program code includes computer readable program code for dynamically associating a first software component with the device driver at run-time. The first software component contains information that facilitates communication with devices of a specific device type. (Specification, page 10, lines 13 - 18; FIG. 6).

Independent Claim 50 is directed to a computer program product for collecting data from a device in which a computer readable storage medium has computer readable program code embodied therein. (Specification, page 4, line 14 - page 5, line 3, and page 9, line 17 - page 10, line 8). The computer readable program code includes computer readable program code for receiving a request to collect data from the device (Specification, page 11, lines 15 - 16; FIG. 9, block 132) and computer readable program code for dynamically associating a software

component with a device driver at run-time. The software component contains information that facilitates communication with the device. (Specification, page 11, lines 21 - 24; FIG. 9, block 134). The computer readable program code further includes computer readable program code for retrieving data from the device using the device driver. (Specification, page 11, lines 24 - 25; FIG. 9, block 136).

Grounds of Rejection to be Reviewed on Appeal

Independent Claims 1, 12, 20, 31, 39, and 50 stand rejected under 35 U.S.C. §102(e) as being anticipated by U. S. Patent No. 6,473,824 to Kreissig et al. (hereinafter "Kreissig").

Argument

I. Introduction to 35 U.S.C. §102 Analysis

Under 35 U.S.C. § 102, "a claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." M.P.E.P. § 2131 (quoting *Verdegaal Bros. v. Union Oil Co.*, 814 F.2d 628, 631, 2 U.S.P.Q.2d 1051, 1053 (Fed. Cir. 1987)). "Anticipation under 35 U.S.C. § 102 requires the disclosure in a single piece of prior art of each and every limitation of a claimed invention." *Apple Computer Inc. v. Articulate Sys. Inc.*, 57 U.S.P.Q.2d 1057, 1061 (Fed. Cir. 2000). "The fact that a certain result or characteristic may occur or be present in the prior art is not sufficient to establish the inherency of that result or characteristic. To establish inherency, the extrinsic evidence 'must make clear that the missing descriptive matter is necessarily present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill. Inherency, however, may not be established by probabilities or possibilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient.'" M.P.E.P. § 2112 (citations omitted).

A finding of anticipation further requires that there must be no difference between the claimed invention and the disclosure of the cited reference as viewed by one of ordinary skill in the art. See *Scripps Clinic & Research Foundation v. Genentech Inc.*, 927 F.2d 1565, 1576, 18 U.S.P.Q.2d 1001, 1010 (Fed. Cir. 1991). In particular, the Court of Appeals for the Federal Circuit held that a finding of anticipation requires absolute identity for each and every element set forth in the claimed invention. See *Trintec Indus. Inc. v. Top-U.S.A. Corp.*, 63 U.S.P.Q.2d

1597 (Fed. Cir. 2002). Additionally, the cited prior art reference must be enabling, thereby placing the allegedly disclosed matter in the possession of the public. *In re Brown*, 329 F.2d 1006, 1011, 141 U.S.P.Q. 245, 249 (C.C.P.A. 1964). Thus, the prior art reference must adequately describe the claimed invention so that a person of ordinary skill in the art could make and use the invention.

Appellants respectfully submit that the pending independent claims are patentable over the cited reference for at least the reason that the cited reference does not disclose or suggest each of the recitations of the independent claims. The patentability of the pending claims is discussed in detail hereinafter.

A. Independent Claims 1, 12, 20, 31, 39, and 50 are Patentable over Kreissig

Independent Claims 1, 12, 20, 31, 39, and 50 stand rejected under 35 U.S.C. §102(e) as being anticipated by Kreissig. Claim 1 is directed to a method of instantiating a device driver and includes the following recitation:

dynamically associating a first software component with the device driver at run-time, **the first software component containing information that facilitates communication with devices of a specific device type.** (Emphasis added).

Claim 12 is directed to a method of collecting data from a device and recites, in part:

...
dynamically associating a software component with a device driver at run-time, **the software component containing information that facilitates communication with the device;**
... (Emphasis added).

Independent Claims 20, 31, 39, and 50 include similar recitations. As indicated above, the pending independent claims describe a software component being associated with a device driver at run-time that contains information that facilitates communication with the device.

Kreissig describes an object-oriented system in which IO domain objects (*see*, FIG. 5, IO domain objects 500 - 506) are dynamically associated with IO Handlers (*see*, FIG. 5, IO Handlers 510 - 516), which represent IO device drivers (Kreissig, col. 8, lines 27 - 29, *see also* col. 9, lines 1 - 12). In sharp contrast to the recitations of the pending independent claims,

however, **the IO domain objects of Kreissig do not contain information that facilitates communication with a device.** Kreissig emphasizes this distinction as follows:

The domain objects must not be confused with the IO device drivers themselves. The methods defined in the domain objects are independent of the interface to the IO device, and they are also independent of the protocol used on said interface. The domain objects only define parameters and boundary conditions for the device's functionality. (Kreissig, col. 8, lines 16 - 21; emphasis added).

According to Kreissig, the IO domain objects that may be associated with IO Handlers at run-time are independent of the interface to the IO device and the protocol associated with the interface. Thus, the IO domain objects cannot contain information that facilitates communication with an IO device because they are designed to be independent of the interface with the IO device and the protocol used to communicate with the IO device.

In response to these arguments, the Second Action cites the passages at col. 2, lines 50 - 57, col. 7, lines 40 - 67, and col. 8, lines 1 - 5, 22 - 27, and 50 - 54 of Kreissig as teaching that a physical object holds a pointer to a device driver and, therefore, the physical object corresponds to the first software component recited in Claim 1 because the physical object can allegedly facilitate communication with devices of a specific type. (Second Action, page 3).

Appellants respectfully disagree with this interpretation of Kreissig's teachings. In particular, Kreissig explains that IO domain object, which is also called an IO physical object (Kreissig, col. 7, lines 58 - 62), communicates with IO handlers by means of an "object reference," which means the IO physical object defines a pointer to the root class of the IO Handlers. (Kreissig, col. 8, lines 50 - 54). Importantly, Kreissig further explains that "[w]ith this approach, it is possible to change the communication path between an IO domain object and its associated device driver at runtime." (Kreissig, col. 9, lines 1 - 3). In sharp contrast to the recitations of the independent claims, however, **the IO domain object or IO physical object is not associated with the driver at run time such that the IO domain object or IO physical object contains information that facilitates communication with devices of a specific device type.** Instead, Kreissig explains that at run time that pointer from the IO domain object or IO physical object can update its pointer to point to a different IO handler and, therefore, access a different device. (Kreissig, col. 9, lines 3 - 11). Thus, Kreissig merely teaches that an IO

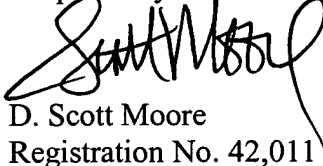
domain object or IO physical object can change its pointer to point to a different IO Handler at run time to access a different device. The IO domain object or IO physical object does not contain any functionality that facilitates communication with devices of a specific type. Instead, the IO Handler is designed to handle communications with a particular device type. The IO domain objects or IO physical objects are independent of the interface to the IO device and the protocol associated with the interface as discussed above. Thus, Kreissig appears to contain no disclosure or suggestion of associating a software component with an IO Handler, which represents a device driver, that facilitates communication with devices of a specific device type as recited in the pending independent claims.

For at least the foregoing reasons, Appellants respectfully submit that independent Claims 1, 12, 20, 31, 39, and 50 are patentable over Kreissig and that Claims 2 - 11, 13 - 15, 21 - 30, 32 - 34, 40 - 49, and 51 - 53 are patentable at least per the patentability of independent Claims 1, 12, 20, 31, 39, and 50. Accordingly, Appellants respectfully request that the rejection of independent Claims 1, 12, 20, 31, 39, and 50 be reversed based on the failure of the Examiner to establish a prima facie case of anticipation under 35 U.S.C. §102 for at least these reasons.

II. Conclusion

In summary, Appellants respectfully submit that, with respect to Claims 1, 12, 20, 31, 39, and 50, the cited reference does not teach all of the recitations of the claims. Accordingly, Appellants respectfully request reversal of the rejection of Claims 1, 12, 20, 31, 39, and 50 based on the cited reference.

Respectfully submitted,



D. Scott Moore
Registration No. 42,011

Myers Bigel Sibley & Sajovec, P.A.
P. O. Box 37428
Raleigh, North Carolina 27627
Telephone: (919) 854-1400
Facsimile: (919) 854-1401
Customer No. 20792

APPENDIX A

1. (Previously presented) A computer implemented method of instantiating a device driver, comprising:

dynamically associating a first software component with the device driver at run-time, the first software component containing information that facilitates communication with devices of a specific device type.

2. (Original) A method as recited in Claim 1, further comprising:
defining a plurality of device parameters;
associating at least one of the plurality of device parameters with a service; and
communicating the at least one of the plurality of device parameters associated with the service to the device driver.

3. (Original) A method as recited in Claim 2, wherein defining the plurality of device parameters comprises:
declaring a parameter base class that defines the plurality of device parameters;
wherein associating the at least one of the plurality of device parameters with the service comprises:
deriving a service-specific sub-class from the base class that defines the at least one of the plurality of device parameters that are associated with the service;
wherein the method further comprises:
instantiating the service-specific sub-class to create a service-specific sub-class object;
and
instantiating the parameter base class to create a parameter base class object.

4. (Original) A method as recited in Claim 3, wherein communicating the at least one of the plurality of device parameters associated with the service to the device driver comprises:
passing the at least one of the plurality of device parameters associated with the service from the service-specific sub-class object to the device driver.

5. (Original) A method as recited in Claim 1, further comprising:
defining a plurality of common device parameters;
defining a plurality of service-specific device parameters;
associating the common device parameters with the service-specific device parameters;
and
communicating the common device parameters and the service-specific device parameters to the device driver.

6. (Original) A method as recited in Claim 5, wherein defining the plurality of common device parameters comprises:
declaring a parameter base class that defines the plurality of common device parameters;
wherein defining the plurality of service-specific device parameters comprises:
providing a second software component that comprises one of a script file and an extensible markup language (XML) file; and
wherein the method further comprises:
instantiating the parameter base class to create a parameter base class object.

7. (Original) A method as recited in Claim 6, wherein associating the common device parameters with the service-specific device parameters comprises:
dynamically loading the parameter base class object with the second software component at run time.

8. (Original) A method as recited in Claim 7, wherein communicating the common device parameters and the service-specific device parameters to the device driver comprises:
passing the common device parameters and the service-specific device parameters from the parameter base class object to the device driver after loading the parameter base class object with the second software component at run time.

9. (Original) A method as recited in Claim 1, wherein the first software component comprises one of a script file and an extensible markup language (XML) file.

10. (Original) A method as recited in Claim 1, wherein dynamically associating the first software component with the device driver at run-time comprises:

selecting the first software component from a plurality of software components, respective ones of the plurality of software components being associated with respective ones of a plurality of device types.

11. (Original) A method as recited in Claim 10, further comprising:
generating the plurality of software components based on a plurality of management information base (MIB) files, respective ones of the plurality of MIB files being associated with respective ones of the plurality of device types.

12. (Previously presented) A computer implemented method of collecting data from a device, comprising:

receiving a request to collect data from the device;
dynamically associating a software component with a device driver at run-time, the software component containing information that facilitates communication with the device; and
retrieving data from the device using the device driver.

13. (Original) A method as recited in Claim 12, wherein retrieving data from the device using the device driver comprises:

associating at least one device parameter with a service;
communicating the at least one device parameter to the device driver; and
retrieving data associated with the at least one device parameter from the device.

14. (Original) A method as recited in Claim 12, wherein the first software component comprises one of a script file and an extensible markup language (XML) file.

15. (Original) A method as recited in Claim 12, wherein dynamically associating the software component with the device driver at run-time comprises:

selecting the first software component from a plurality of software components, respective ones of the plurality of software components being associated with respective ones of a plurality of device types.

16 - 19. (Canceled)

20. (Original) A system for instantiating a device driver, comprising:
means for dynamically associating a first software component with the device driver at run-time, the first software component containing information that facilitates communication with devices of a specific device type.

21. (Original) A system as recited in Claim 20, further comprising:
means for defining a plurality of device parameters;
means for associating at least one of the plurality of device parameters with a service; and
means for communicating the at least one of the plurality of device parameters associated with the service to the device driver.

22. (Original) A system as recited in Claim 21, wherein the means for defining the plurality of device parameters comprises:
means for declaring a parameter base class that defines the plurality of device parameters;
wherein the means for associating the at least one of the plurality of device parameters with the service comprises:
means for deriving a service-specific sub-class from the base class that defines the at least one of the plurality of device parameters that are associated with the service;
wherein the system further comprises:
means for instantiating the service-specific sub-class to create a service-specific sub-class object; and
means for instantiating the parameter base class to create a parameter base class object.

23. (Original) A system as recited in Claim 22, wherein the means for communicating the at least one of the plurality of device parameters associated with the service to the device driver comprises:

means for passing the at least one of the plurality of device parameters associated with the service from the service-specific sub-class object to the device driver.

24. (Original) A system as recited in Claim 20, further comprising:

means for defining a plurality of common device parameters;

means for defining a plurality of service-specific device parameters;

means for associating the common device parameters with the service-specific device parameters; and

means for communicating the common device parameters and the service-specific device parameters to the device driver.

25. (Original) A system as recited in Claim 24, wherein the means for defining the plurality of common device parameters comprises:

means for declaring a parameter base class that defines the plurality of common device parameters;

wherein the means for defining the plurality of service-specific device parameters comprises:

means for providing a second software component that comprises one of a script file and an extensible markup language (XML) file; and

wherein the system further comprises:

means for instantiating the parameter base class to create a parameter base class object.

26. (Original) A system as recited in Claim 25, wherein the means for associating the common device parameters with the service-specific device parameters comprises:

means for dynamically loading the parameter base class object with the second software component at run time.

27. (Original) A system as recited in Claim 26, wherein the means for communicating the common device parameters and the service-specific device parameters to the device driver comprises:

means for passing the common device parameters and the service-specific device parameters from the parameter base class object to the device driver after loading the parameter base class object with the second software component at run time.

28. (Original) A system as recited in Claim 20, wherein the first software component comprises one of a script file and an extensible markup language (XML) file.

29. (Original) A system as recited in Claim 20, wherein the means for dynamically associating the first software component with the device driver at run-time comprises:

means for selecting the first software component from a plurality of software components, respective ones of the plurality of software components being associated with respective ones of a plurality of device types.

30. (Original) A system as recited in Claim 29, further comprising:

means for generating the plurality of software components based on a plurality of management information base (MIB) files, respective ones of the plurality of MIB files being associated with respective ones of the plurality of device types.

31. (Original) A system for collecting data from a device, comprising:

means for receiving a request to collect data from the device;

means for dynamically associating a software component with a device driver at run-time, the software component containing information that facilitates communication with the device; and

means for retrieving data from the device using the device driver.

32. (Original) A system as recited in Claim 31, wherein the means for retrieving data from the device using the device driver comprises:

- means for associating at least one device parameter with a service;
- means for communicating the at least one device parameter to the device driver; and
- means for retrieving data associated with the at least one device parameter from the device.

33. (Original) A system as recited in Claim 31, wherein the first software component comprises one of a script file and an extensible markup language (XML) file.

34. (Original) A system as recited in Claim 31, wherein the means for dynamically associating the software component with the device driver at run-time comprises:

- means for selecting the first software component from a plurality of software components, respective ones of the plurality of software components being associated with respective ones of a plurality of device types.

35 - 38. (Canceled)

39. (Original) A computer program product for instantiating a device driver, comprising:

- a computer readable storage medium having computer readable program code embodied therein, the computer readable program code comprising:

- computer readable program code for dynamically associating a first software component with the device driver at run-time, the first software component containing information that facilitates communication with devices of a specific device type.

40. (Original) A computer program product as recited in Claim 39, further comprising:

- computer readable program code for defining a plurality of device parameters;

computer readable program code for associating at least one of the plurality of device parameters with a service; and

computer readable program code for communicating the at least one of the plurality of device parameters associated with the service to the device driver.

41. (Original) A computer program product as recited in Claim 40, wherein the computer readable program code for defining the plurality of device parameters comprises:

computer readable program code for declaring a parameter base class that defines the plurality of device parameters;

wherein the computer readable program code for associating the at least one of the plurality of device parameters with the service comprises:

computer readable program code for deriving a service-specific sub-class from the base class that defines the at least one of the plurality of device parameters that are associated with the service;

wherein the computer program product further comprises:

computer readable program code for instantiating the service-specific sub-class to create a service-specific sub-class object; and

computer readable program code for instantiating the parameter base class to create a parameter base class object.

42. (Original) A computer program product as recited in Claim 41, wherein the computer readable program code for communicating the at least one of the plurality of device parameters associated with the service to the device driver comprises:

computer readable program code for passing the at least one of the plurality of device parameters associated with the service from the service-specific sub-class object to the device driver.

43. (Original) A computer program product as recited in Claim 39, further comprising:

computer readable program code for defining a plurality of common device parameters;

computer readable program code for defining a plurality of service-specific device parameters;

computer readable program code for associating the common device parameters with the service-specific device parameters; and

computer readable program code for communicating the common device parameters and the service-specific device parameters to the device driver.

44. (Original) A computer program product as recited in Claim 43, wherein the computer readable program code for defining the plurality of common device parameters comprises:

computer readable program code for declaring a parameter base class that defines the plurality of common device parameters;

wherein the computer readable program code for defining the plurality of service-specific device parameters comprises:

computer readable program code for providing a second software component that comprises one of a script file and an extensible markup language (XML) file; and

wherein the computer program product further comprises:

computer readable program code for instantiating the parameter base class to create a parameter base class object.

45. (Original) A computer program product as recited in Claim 44, wherein the computer readable program code for associating the common device parameters with the service-specific device parameters comprises:

computer readable program code for dynamically loading the parameter base class object with the second software component at run time.

46. (Original) A computer program product as recited in Claim 45, wherein the computer readable program code for communicating the common device parameters and the service-specific device parameters to the device driver comprises:

computer readable program code for passing the common device parameters and the service-specific device parameters from the parameter base class object to the device driver after loading the parameter base class object with the second software component at run time.

47. (Original) A computer program product as recited in Claim 39, wherein the first software component comprises one of a script file and an extensible markup language (XML) file.

48. (Original) A computer program product as recited in Claim 39, wherein the computer readable program code for dynamically associating the first software component with the device driver at run-time comprises:

computer readable program code for selecting the first software component from a plurality of software components, respective ones of the plurality of software components being associated with respective ones of a plurality of device types.

49. (Original) A computer program product as recited in Claim 48, further comprising:

computer readable program code for generating the plurality of software components based on a plurality of management information base (MIB) files, respective ones of the plurality of MIB files being associated with respective ones of the plurality of device types.

50. (Original) A computer program product for collecting data from a device, comprising:

a computer readable storage medium having computer readable program code embodied therein, the computer readable program code comprising:

computer readable program code for receiving a request to collect data from the device;
computer readable program code for dynamically associating a software component with a device driver at run-time, the software component containing information that facilitates communication with the device; and

computer readable program code for retrieving data from the device using the device driver.

51. (Original) A computer program product as recited in Claim 50, wherein the computer readable program code for retrieving data from the device using the device driver comprises:

computer readable program code for associating at least one device parameter with a service;

computer readable program code for communicating the at least one device parameter to the device driver; and

computer readable program code for retrieving data associated with the at least one device parameter from the device.

52. (Original) A computer program product as recited in Claim 50, wherein the first software component comprises one of a script file and an extensible markup language (XML) file.

53. (Original) A computer program product as recited in Claim 50, wherein the computer readable program code for dynamically associating the software component with the device driver at run-time comprises:

computer readable program code for selecting the first software component from a plurality of software components, respective ones of the plurality of software components being associated with respective ones of a plurality of device types.

54 - 57. (Canceled)

In re: Tabares et al.

Serial No.: 09/992,155

Page 19

APPENDIX B – EVIDENCE APPENDIX

None

In re: Tabares et al.

Serial No.: 09/992,155

Page 20

APPENDIX C – RELATED PROCEEDINGS APPENDIX

None.